International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 05, May-2015

From Relational Database to Column-Oriented NoSQL Database: Migration Process

Steve Ataky Tsham Mpinda, Luís Gustavo Maschietto,
Departamento de Computação
Universidade Federal de São Carlos
São Carlos, Brazil

Patrick Andjasubu Bungama
Departamento de Computação
Universidade Federal do Paraná
Curitiba, Brazil

***Abstract*—Wherever there is a need for the availability of services with a high traffic, maintaining at the same time a high performance, the blocking point often is the database. When one talks about high traffic systems or services and databases, one rarely hears of relational databases. Indeed, to ensure data consistency is expensive in time and is often inconsistent with performance. Since the relational model does not seem suitable in environments requiring large architectures and the ACID properties of bases generally do not allow to scale, a new movement was born from the initiative of cloud's architects and other communities like Amazon and LinkedIn: NoSQL (aka: Not Only SQL). NoSQL databases are radically changing the architecture of the database that one used to see and thus allow to increase the performance and availability of services. Thereby, it seems useful to put forward a migration approach of conventional database to a NoSQL database. To do this, some limitations of relational RDBMS to dealing with large volumes of data is presented. Then the NoSQL technology and its strengths, issues and challenges are discussed. Finally, a migration from a relational database to column oriented NoSQL database was proposed as the aim of this paper.**

Keywords—NoSQL, Migration, column oriented database, data conversion, schema translation, column family, super column.

I. INTRODUCTION

Since the storage systems and data manipulation used so far cannot reasonably meet the requirements such as web service availability and bet and face scalability resulting, the exponential growth of data, taking into account weakly structured data and technological advances, companies are gradually and inevitably, at least for those who wish to remain competitive, forced to migrate their IT systems to new architectures. Thereof, the migration of conventional Database Management Systems to new NoSQL type databases, that is to say the data engines that do not use the SQL standard. NoSQL, in turn, offer better data availability, huge storage capacities by overcoming constraints induced by the ACID properties. They are widely used and have gotten a place in the IT infrastructure.

Given the maturity and the good reputation conventional databases enjoy, the concern herein is to find out what the new databases offer to supporting the migration from relational to NoSQL. In other words, what are the limits of relational systems regarding the availability constraints (for scalability) which are the subject of most services, as well as the increasing volume of data?

Taking into account the present situation, as part of this work will be presented an approach migration of a relational database to a database which provides a better scalability and improved flexibility. In other words, it is to propose a transitional approach of relational databases to columns oriented databases (NoSQL). To do this, firstly was presented the reasons for migrate; thereof there is a need for presenting the limitations of relational databases in a distributed environment with high traffic and high data volume Context. The reasons for the choice of NoSQL databases over another type of databases are also discussed. After all, it will be presented a case study-based migration.

II. THE LIMITATIONS OF RDBMS

In addition of the relational model, most relational RDBMS are transactional, which imposes compliance constraints Atomicity, Consistency, Isolation and Durability, commonly called ACID properties in short [2]. In a centralized context, these constraints are rather easy to guarantee. In the case of distributed systems, it is however necessary to distribute the processing and data between different servers. At this point, it then becomes difficult to maintain ACID constraints across the entire distributed system while maintaining good performance. Despite this, it is not obvious to overcome the stranglehold of relational databases on data management matters. Yet it is to this that works a group of actors recent years highlighting the limitations of the relational model for some types of distributed applications with high traffic and dealing with large volumes of data [3][8].

Notwithstanding, in terms of data management, a new technical approach just tickles the dominant model every twenty years or so. At first, in the 70s, the relational was one of them, replacing the different models in place (hierarchical, network, etc.) without discussion. It was adapted, extended and its simple ubiquity has marginalized ambitious despite their undeniable arguments [3]. In a way, this movement embodies the return of concepts put under the extinguisher for many years. This is a strong tendency; applications Business operate on data volumetric increasingly large. Moreover, a plethora of users and customers access to such data, either directly or, more often, indirectly. And this is clearly not the SOA or management of reference data (master data management) that will influence these trends! As result, a pressure of growing exercised over conventional systems data base management. Faced with this phenomenon, brute force can it be enough? [4]

- more Mhz
- more processors,
- more memory ...
- But not always at much reasonable cost!

But it would not count another consequence of the volume of data: the administration bases of large data is demanding. Thus, administering more than one database tens (or even hundreds) of millions of records such as a few tens of thousands of records. With such volumes of data, once the brief operations can take hours as modifying the schema or the removal of a significant number of records. The emergence of these new paradigms has led to major changes in the approach to the application design and their relationship to the database. When speaking of high traffic sites and databases, one rarely hears of relational databases. Indeed, ensuring data consistency is expensive in time and is often incompatible with the performance. Since the relational model does not seem suitable in environments requiring large architectures, the ACID properties of bases generally do not allow to scale, a new movement was born from the initiative of the architects of Cloud Computing and community sites like Facebook, Amazon and LinkedIn: NoSQL. [4]

Moreover, the relational database's properties even though are necessary to the logic of relational, however highly harm the performance, especially the property of consistency. Indeed, consistency is very difficult to implement in the context of multiple servers (distributed environment), because to do so, all servers must be mirrors of each other, thus two problems arise:

- The storage cost is enormous because each data is present on each server
- The cost of inserting, altering and deleting is greater, as one cannot commit a transaction unless to be sure that it was performed on all servers and the system makes the user wait during this time [10].

In the Figure 1 below, the performed updates on a server should be passed on to other servers for the system remains consistent. All the servers: s1, s2, s3 and s4 have to have the same copy to the database.

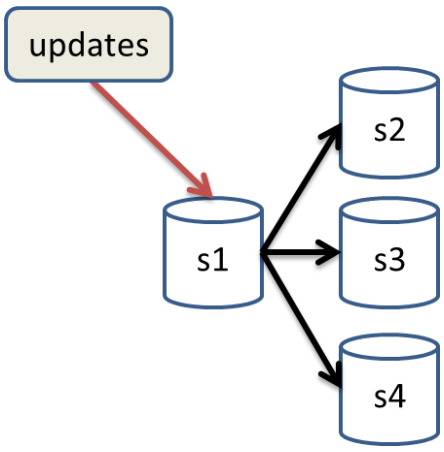

Figure 1: ACID properties problem in a distributed environment.

III. CURRENT POSITION OF NOSQL

The NoSQL movement reached its heyday in recent years, particularly as it seeks to address several issues that relational databases do not respond adequately [9]:

- availability to handle very large volumes and Partitioning;
- flexibility scheme;
- difficulty to represent and process complex structures such as trees, graphs, or relationships in large numbers (In the databases ecosystem, graphs bases are often positioned mainly in the last two points:);
- process highly connected data;
- easily manage a complex and flexible model;
- deliver outstanding performance for local readings, for graph traversal;

IV. NOSQL, OTHER APPROACH TO STORAGE AND MANIPULATE DATAS

NoSQL or "Not Only SQL" is a recent movement (2009), concerning the databases. The idea of movement is simple: to offer alternatives to relational databases to stick to new trends and architectures of time, including cloud computing. The main points of NoSQL are high availability and horizontal data partitioning, to the detriment of consistency. While current relational databases are based on the ACID properties (Atomicity, Consistency, Isolation and Durability). NoSQL means "Not Only SQL"; this term refers to all the databases that are opposed to the concept of relational DBMS.

The definition, "not only SQL," provides an initial response to the question "will the NoSQL kill the relational databases?". Indeed, NoSQL does not replace relational database but offers an alternative or supplement of the functionality of the RDBMS to provide more interesting solutions in some contexts. NoSQL includes many databases, mostly recent, differentiated model with SQL by logic of non-relational data representation. Their main advantages are their performance and their ability to handle very large volumes of data. However, in the projects, one should not oppose both approaches but often makes them coexist! This technology (NoSQL) ultimately aims not to replace traditional DBMS but rather to complement them by deporting some of the burden of processing and data storage to third-servers (web in conventional architectures)

NoSQL databases have often emerged as internal projects of large commercial sites or research. Thus, BigTable, whose development began in 2004, manages most of the data from Google, almost absolute reference in terms of volume and performance. Several major sites use these particular tools: Cassandra Facebook and Dynamo at Amazon, PNUTS at Yahoo, etc. And the current orientation Cloud Computing only reinforces the interest for these approaches, as evidenced by the storage services of the Microsoft Azure platform [3].
In the wake of these proprietary solutions, many open source projects have emerged. Although a diversity reign, several common features seem to emerge: lack of scheme, horizontal

partitioning on a large number of nodes, de-normalization, automatic replication, versioning of data, and so on. For sure, all this comes to demolish a range of principles built in dogma for over thirty years. As we have seen, most of the movement NoSQL DBMS has been constructed by ignoring the constraints ACID, even not to offer functionality transactional. NoSQL group aims primarily to demonstrate that there are now alternatives solutions used in demanding systems. The purpose is obviously to obtain a better data availability, through extensive partitioning capabilities but at a stress release of ACID properties.

Types of NoSQL databases

There are four basic categories of NoSQL. [1]

- Oriented columns, such as HBase, Cassandra Hypertable or they are based on the concept of Google's BigTable
- Based on graph theory (Euler) implemented by Neo4j.
- Oriented key-value (Voldemort, Dynomite, Riak).
- Oriented document, such as CouchDB.

Since the work aim the oriented column, one will present its specificities in more detail at the next point.

V. COLUMN-ORIENTED DATABASES

While columns are static for a relational database, they are dynamic for a column-oriented database, it is then possible to add columns dynamically and there is no storage cost for nulls.

For performance reasons the columns are sorted on the disk, minimizing random access. In addition to Cassandra, for instance, sequential writes are to avoid hard disk latencies, data is first written to memory, and then persisted to disk during a commit log or when the memory is full.

BDD oriented columns are provided for storing millions of columns, which makes it suitable bases storing one-to-many. The disadvantage is the update. While for a relational database, an update of tuples with a foreign key can be enough, a column-oriented BDD may require an update of all values in a column for all records. There are also great columns which are columns of containers.

The query is quite minimalist, for example:

- All columns which key 256
- The column name is from 'bbb' and 'bbc' and whose key is 8652
- The column 'abc' for lines ranging from 500 to 1000.

Cassandra (originally used by Facebook for non instant messages) and HBase are solutions BDD oriented columns. Cassandra allowed Facebook to access messages exchanged between users and messages containing certain words. These bases are for uses where one must store data per user unique data. Do not look to the relationship with this basis, the mailing list is full of such questions but it's just not done!

Column-oriented databases are also an evolution of key-value model[1]. Originally, this model was developed by Google to BigTable in which data are stored according to a column-oriented model. Instead of storing data in tables in rows / columns as in most of the RDBMS, data is distributed here in dynamic columns can they even contain one or more values. Each line of data and having a different number of columns and each column can contain a different number of values. The column-oriented model has the advantage of improving storage efficiency and avoids eating space compared to conventional RDBMS table model. Indeed, because of their design by allocating blocks in an RDBMS, an empty column still consume space. Moreover, this model allows at any time using a new column, and we gain extensibility at the data schema. The Figure 2 [1] illustrate the records storage example in a column-oriented database.

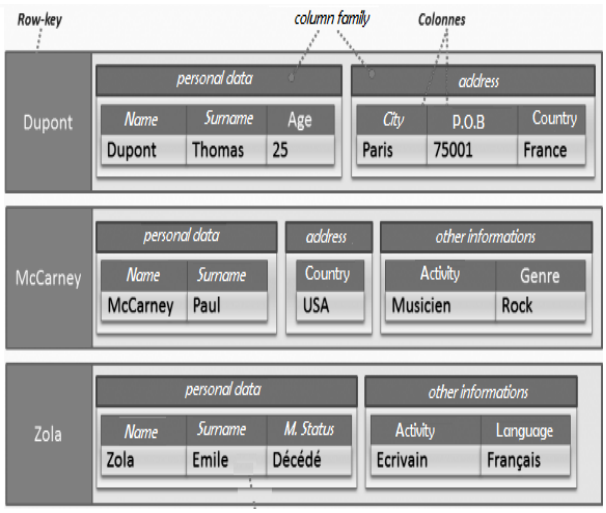

Figure 2: Example of the records storage example in a column-oriented database. Adapted from [1]

VI. MIGRATION APPROACH OF A RELATIONAL DATABASE TO NOSQL

1. Source data model

In a migration process and more exactly to the schema translation phase it normally used as the source data schema the physical data model (PDM) that is to say the script of the relational database. Nonetheless, it can vary from RDBMS to another, even if the standard language remains the SQL. Furthermore, the relational data model, introduced by Codd [1970] represents a database as a collection of relations whereof the database name relational [6].

2. Target data model

A column-oriented database can be seen as a complex data structure having at least 5 dimensions based on concepts such as: tables, columns families, key, super columns and simple columns. RDBMS are oriented lines. This means that all the columns in a row (entry identified by a key) are stored one after another, in an aggregated manner. A column-oriented database, meanwhile, stores data by focusing on data grouping entire columns.

3. Translation of the source data model into target model

A data schema S1 of a relational database can be translated into an equivalent target schema, S2 expressed as a column-oriented model through a set of translation rules applications.
The generation of the target schema depends on the flexibility of the rules. Each rule reflects a specific construction. The two equivalent diagrams should take the overall semantics, as well as situational.

Algorithm 1 Data migration algorithm

```
1: Build a relational dependency matrix of the source database
2: From the target data model build the key mapping table.
3: for Any table in the dependency matrix do
4:     Take an element (a table)
5:     extract the set of tuples of the table (select * from table name)
6:     for every single selection line do
7:         Take a line or tuple selection
8:         Instantiate the equivalent object in the target database
9:         Add a new entry in the keys mapping table with the key of the
   processing line and the identifier of the newly instantiated object
10:        Assign values to the attributes of the instantiated object with the
   fields of the processing tuple.
11:        Search possibly the parent object
12:        Update the parent column of the processing object
13:        Update of the list of the son of the parent object.
14:        move to the next row or tuple selection.
15:    end for
16:    move to the next table in the dependency matrix
17: end for
```

The herein approach proposes to receive as input an existing relational database and generates an equivalent data model that reflects the essential characteristics for the NoSQL database. The approach should not only generate the target schema: the translation patterns. But also to generate the data instances to the target database from the starting database. The entry model is the Entity Association model while the destination is a kind of class diagram. That is to say, generic statements or definitions objects. This is in fact the way the objects are stored.

Concerning the data conversion, it comes in three stages: data extraction, data processing or data enrichment and injection in the destination database. The data extraction consists essentially to querying a relational database through operations and tools that are usual such as SQL commands and the use of joints. Data processing, in turn, involves the transformation of data from the target database format so that future data injection is done without major difficulties. On the data integration, it is needed to instantiate and initiate the object classes, that is to say the various data structures of a column-oriented database. This is done in the right order columns, super columns, and families of columns. The data conversion phase introduces the concepts like the definition of a key management policy and the need for defining a mapping table keys and the construction of a relational dependency matrix. All these aspects helped to propose the data conversion algorithm below {algorithm 1}.

With regard to the statement 10, it concerns only the tables with at least one foreign key. Super columns, in turn, will be fulfilled by the son objects during processing. Any object with a parent will update its parent corresponding attribute and then add the reference or the object (depending on desired management) in the super column (list of son objects). The instruction 8 applies only to tables in the source database that have an equivalent in the column family. That is to say, if a table does not appear in the key mapping table, the statement will be applied to it when migrating data from this table. The statement 7, 8 and 9 refer to the processing and to the insertion of data in the target database. The statement 9 requires for its execution that the key mapping table is provided earlier. The statements 8 and 9 can be swapped without having any impact on the algorithm.

To convert data or instances of data from a relational database, it is required to extracted and injected into the new non-relational database. To carry out this task, it is necessary to ensure that all of the data has been transferred and the filled database is consistent. To do so, we will rely on the developed data mapping algorithm above.

VII. CONCLUSION

Taking into account faults or problems with relational Databases, it has been established that despite their maturity, their overwhelming use, and their ubiquitous presence, relational databases meet some limits based on immersive environments. In a centralized environment, it is reproached their low level of flexibility facing the management of heterogeneous objects. Indeed, all the attributes of an object must be declared and known in advance in the relational world, what, in turn, does not offer these great scalability factor. The other major limitation of relational databases in centralized environment is the use of joins, which is not always evident in great or big volume. In other words, requests using the joins are not always optimal and cause increasingly long latency time in case of exponential increase in data.

Regarding the limitations of relational databases in a distributed environment, one can mention the difficult applicability of ACID properties, including consistency which requires the use of equipment increasingly specialized and therefore expensive (over MHZ, more memory, etc.). Moreover, the CAP theorem shows sufficiently the existence of a different approach of storage and data manipulation which includes the AP and PC systems in the NoSQL movement, which coexists with the AC system, that is, relational and transactional.

In terms of the NoSQL technology, it is a new approach of storage and data manipulation which advocates abandoning ACID principles in favor of concepts such as horizontal data partitioning that enables data sharing and processing across multiple servers and flexibility of data schema to effectively manage weakly structured data. For now, NoSQL is a growing movement, although it is used by major accounts that are at the origin of the movement. There are no solutions that really stand out of the lot, there are only solutions adapted to the needs. NoSQL databases are used to make the system much more efficient and resistant to failure, however as these do not always provide data consistency, they are rarely used alone: a relational database will contain information where consistency is vital and a NoSQL database will contain all the rest.

Since the aim of this paper is the proposal of a migration approach from a relational database to a column-oriented database (NoSQL), it was determined the causes, the shortcomings of relational database, which would justify such an operation as well as the strengths of NoSQL databases that motivate migration of the first to the last. Then a migration approach was discussed and a migration algorithm.

REFERENCES

(1) http://davidmasclet.gisgraphy.com/post/2010/06/09/10-minutes-pour-comprendre...NoSQL
(2) Avinash Lakshman, Prashant Malik, Structured Storage System over a P2P Network
(3) http://www.zdnet.fr/blogs/codes-et-modeles/nosql-une-reponse-aux-limites-du-relationnel-39711127.htm,Mars 2012
(4) Nicolas Dasriaux, Bases de données à haute volumétrie : comment faire.
(5) Michaël Figuière, Bases de données orientées colonnes et Cassandra, Mai 2010.
(6) Abdelsalam Amraga Maatuk, migrating relational databases into object-based and xml databases.
(7) Shalini Batra, Charu Tyagi ,"Comparative Analysis of Relational And Graph Databases" International Journal of Soft Computing and Engineering (IJSCE) Volume-2, Issue-2, May 2012 ,pp-509-512.
(8) J. Fong, H.K. Wong, Z. Cheng ,"Converting relational database into XML documents with DOM" Information and Software Technology 45(2003)335 –355.
(9) Mpinda, Steve Ataky T.; Maschietto, Gustavo L.; Bungama, Andjasubu P., "Graph DataBase Application Using Neo4j (railtoad simulation)", *International Journal of Engeneering Research and Technology*, vol.4-Issue 04,April-2015.
(10) Mathieu Roger, synthèse d'étude et projets d'intergiciels: bases NoSQL, 2010